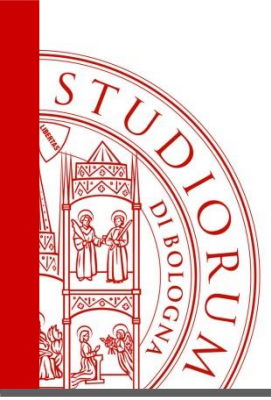


Università degli Studi di Bologna
Corso di Laurea in Ingegneria Informatica

Diagramma dei Componenti e di Deployment

Ingegneria del Software T

Prof. MARCO PATELLA
Dipartimento di Informatica – Scienza e Ingegneria (DISI)

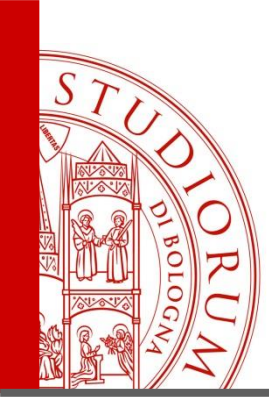


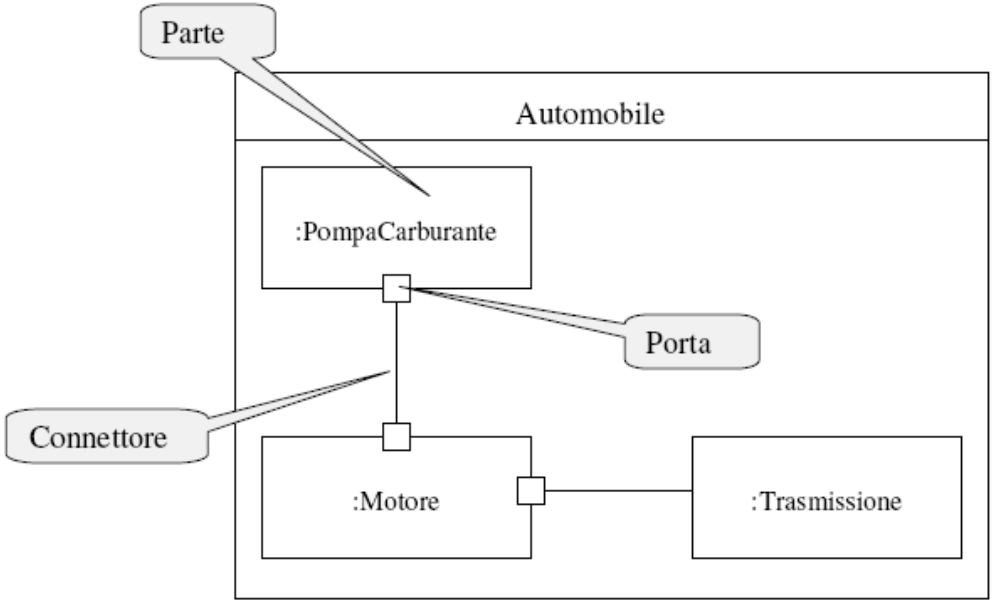
Diagramma dei componenti

Diagramma dei Componenti

- Da UML 2.0 il concetto di componente si è evoluto rispetto alla versione precedente dello standard
- Specifica un *contratto formale di servizi offerti e richiesti* in termini di interfacce (eventualmente esposte tramite porte)
- Il concetto di componente è strettamente legato a quello di ***struttura composita*** che spesso viene impiegata per rappresentare le parti interne del componente
- Un componente è tipicamente specificato da uno o più classificatori (ad es. classi) e può essere implementato da uno o più *artefatti* (file eseguibile, script, …)
- Gli **internals** (parti interne) sono inaccessibili solo attraverso le interfacce

Struttura Composita

- Il Diagramma di Struttura Composita ha l'obiettivo di rappresentare la struttura interna (le parti) di un classificatore (classe, componente…), inclusi i punti di interazione (porte) utilizzati per accedere alle caratteristiche della struttura



Struttura Composita

- Introdotto per scomporre gerarchicamente un classificatore, mostrandone la struttura interna:
  - mostra la struttura interna di un classificatore complesso
  - mostra in modo separato l'***interfaccia*** di un classificatore dalla sua ***struttura interna***
  - descrive i ruoli che i diversi elementi della struttura giocano per soddisfare l'obiettivo della struttura stessa e le interazioni richieste
- Questo permette al progettista di prendere un oggetto complesso e spezzarlo in parti più piccole e semplici
- È una sorta di strumento di zoom utile per gestire la complessità di rappresentazione

Package vs. Struttura Composita

- Per capire bene la differenza tra i package e le strutture composite bisogna pensare che
  - i primi rappresentano un raggruppamento logico al momento dell'analisi
  - mentre le seconde fanno riferimento a quello che succede durante l'esecuzione
- Di conseguenza le strutture composite sono adatte a rappresentare i componenti e le loro parti, e sono usate spesso nei diagrammi dei componenti

Diagramma dei Componenti

- Da UML 2 è una specializzazione della metaclasse Class
- Quindi un componente può avere attributi e metodi, una struttura interna, porte e connettori
- Da UML 2 l'icona del componente è cambiata

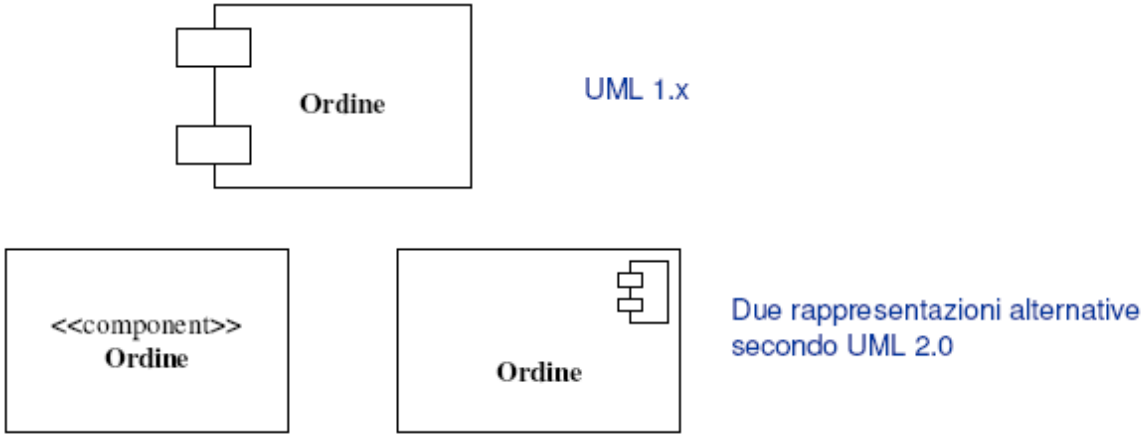
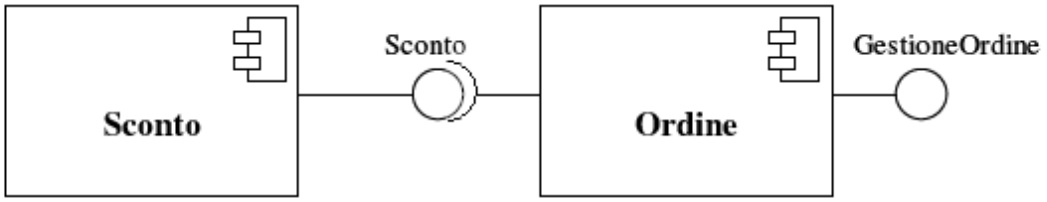
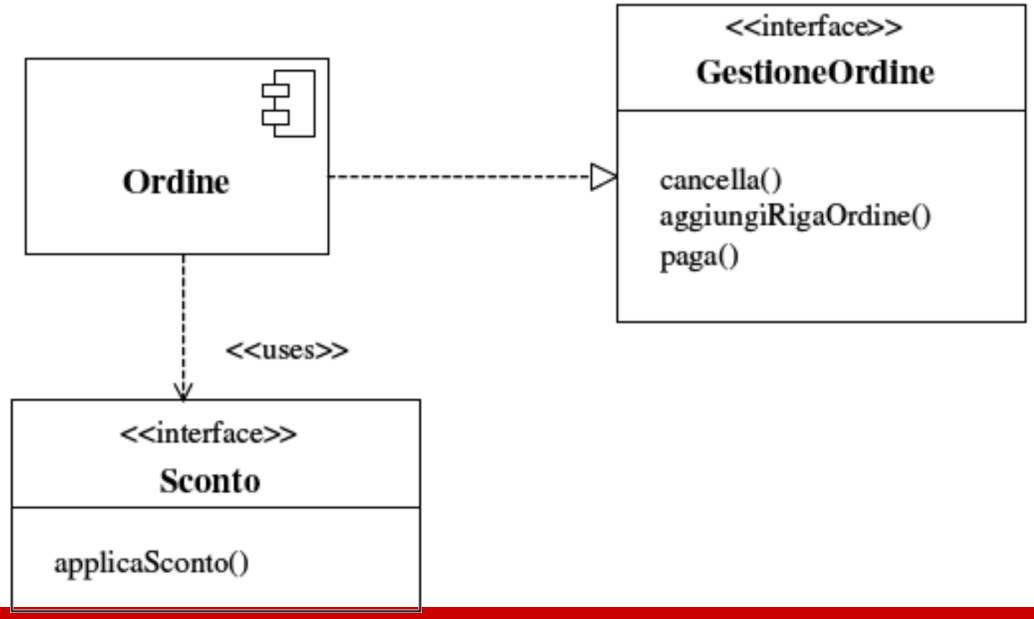


Diagramma dei Componenti

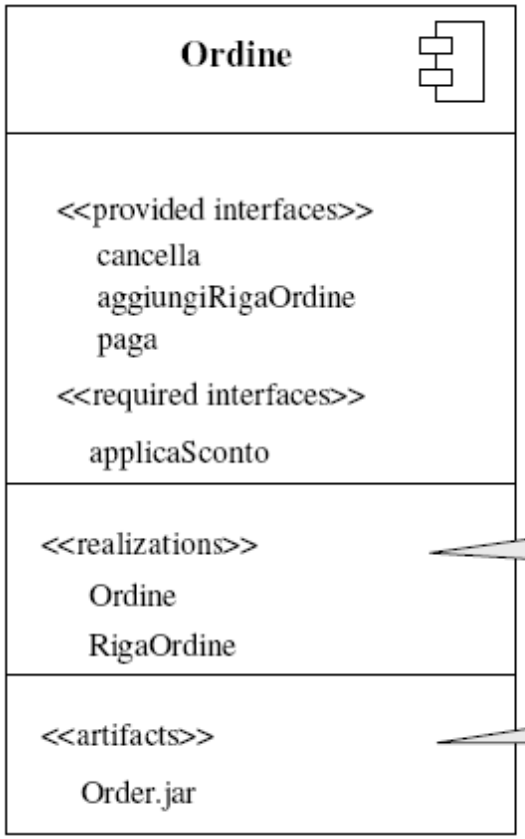
Relazioni tra componenti



- Interfaccia fornita e interfaccia richiesta devono essere compatibili a livello di tipo (attributi e associazioni) e di vincoli sul comportamento (operazioni, eventi)



Componenti: White-box



La notazione è detta white-box, perché mostra le proprietà interne del componente

sono le classi o i componenti che lo costituiscono

sono gli artefatti che lo implementano

Diagramma dei componenti

- I classificatori interni (internals) che realizzano un componente possono essere mostrati in due modi:
  - innestati nel componente

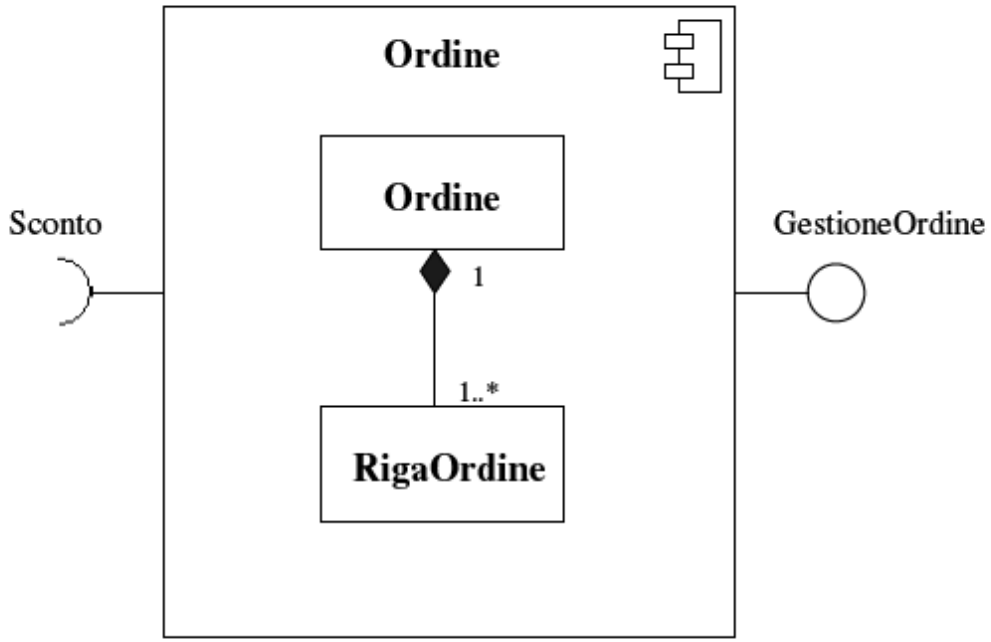
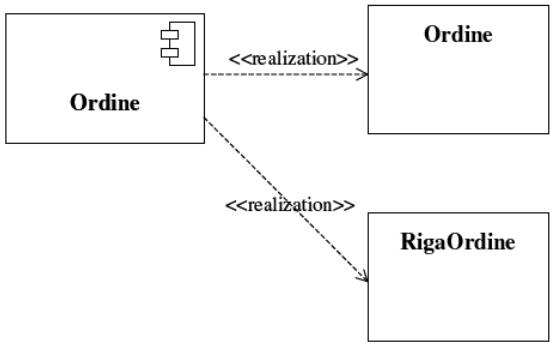


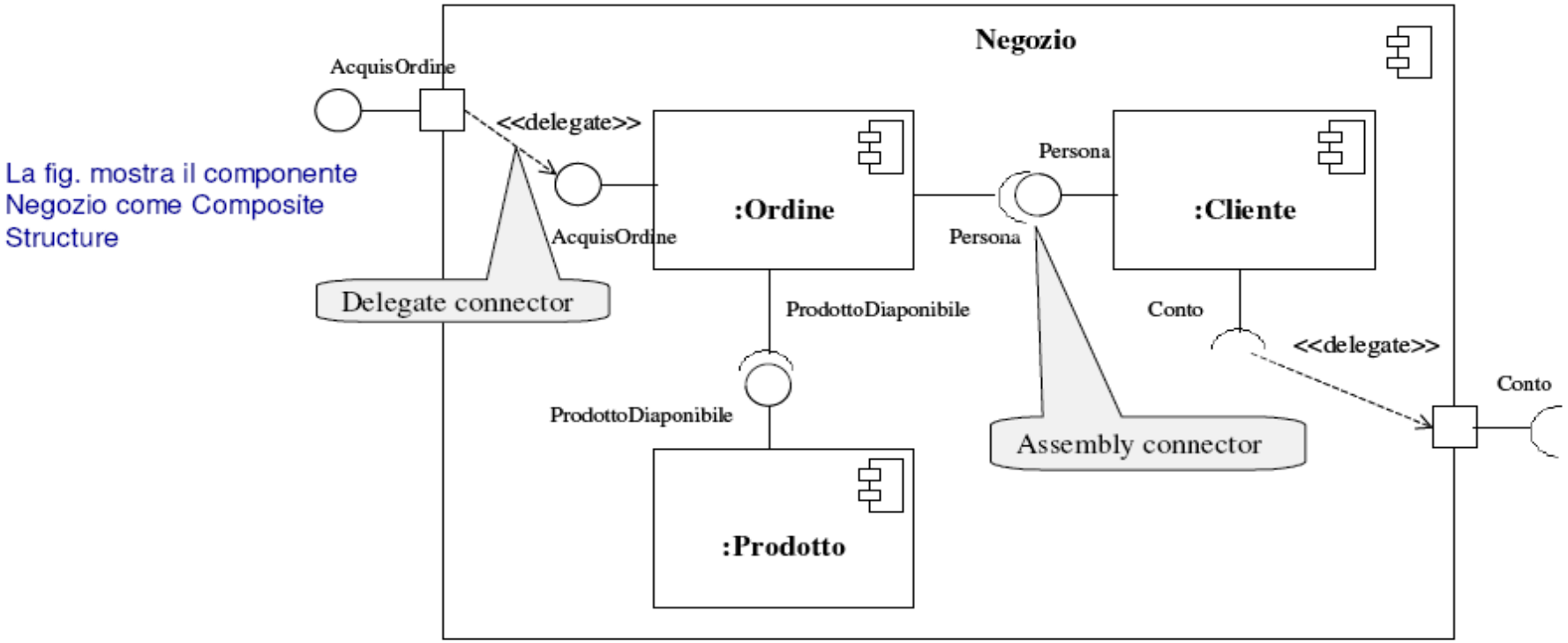
Diagramma dei componenti

- In modo esplicito tramite la dipendenza di **realization**



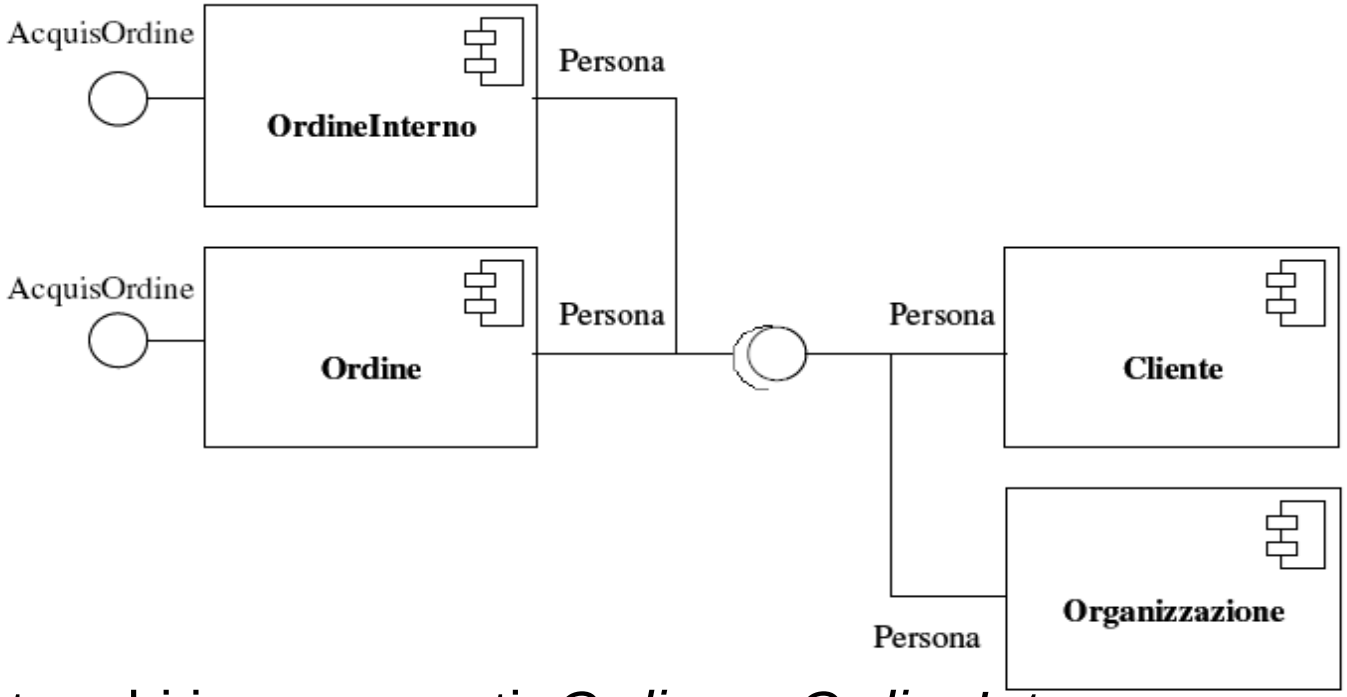
- Il componente *Ordine* è implementato istanziando le classi *Ordine* e *RigaOrdine*
- La ***realization*** è una dipendenza specializzata tra due insiemi di elementi di modellazione, di cui uno rappresenta la specifica e l'altro una sua implementazione
- Per un componente la **realization** definisce i classificatori che realizzano il contratto offerto dal componente stesso in termini delle sue interfacce offerte e richieste

Diagramma dei componenti



- Le parti interne sono collegate direttamente tra loro (assembly connector) oppure connesse a porte sul confine del componente (delegated connector)
- I delegated connector sono utilizzati per esporre servizi di una “parte” all’esterno del container

Connettori Multiple Wiring



- Entrambi i componenti, *Ordine* e *OrdineInterno*, richiedono l'interfaccia *Persona*: l'applicazione non conosce, fino al momento dell'esecuzione, quale componente, *Cliente* o *Organizzazione*, fornirà il servizio richiesto
- Si tratta di ***un'interazione polimorfica***

Connettori Multiple Wiring

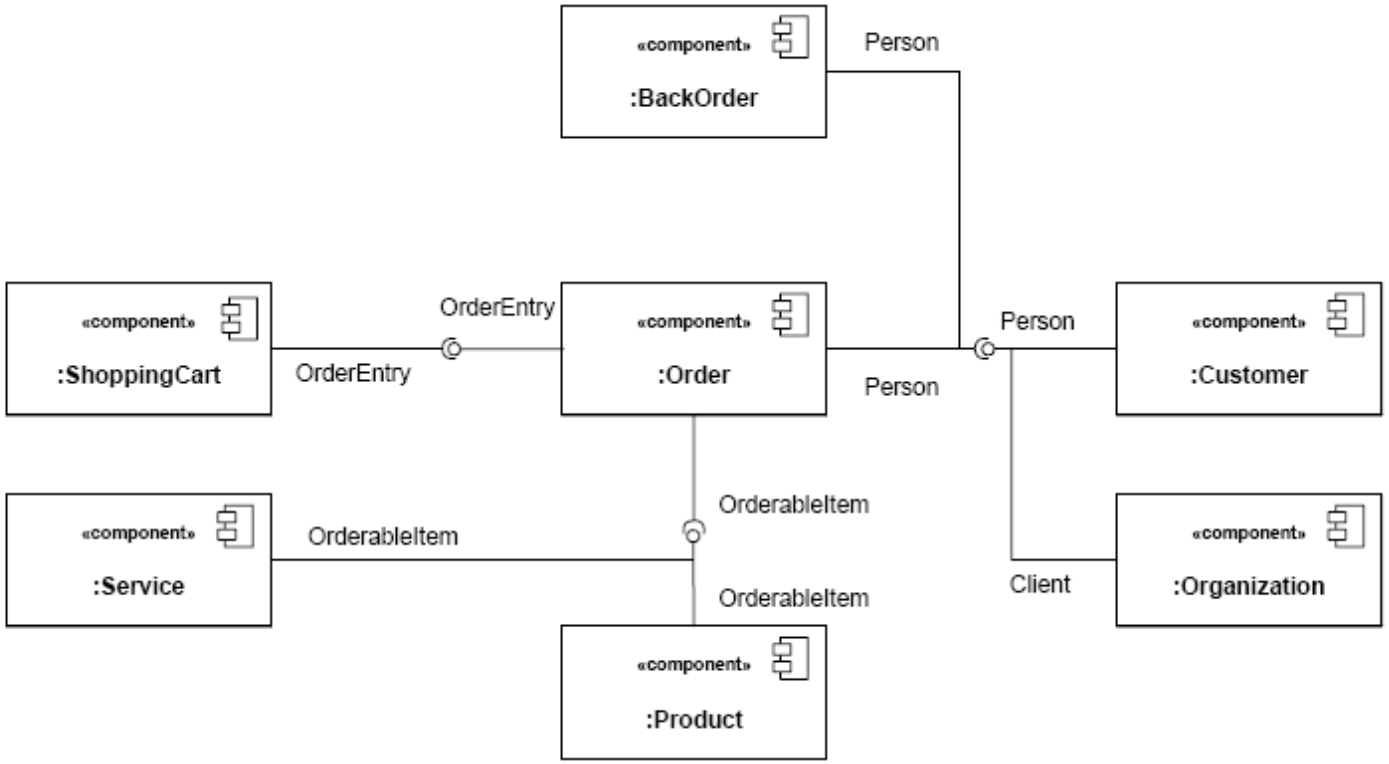
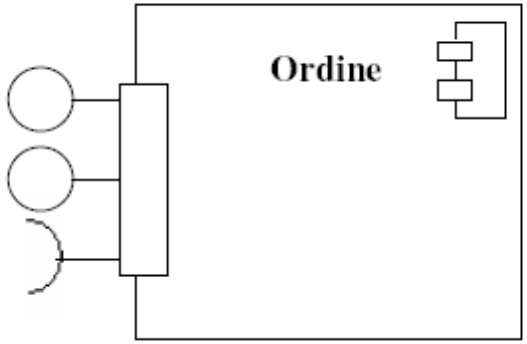


Diagramma dei componenti

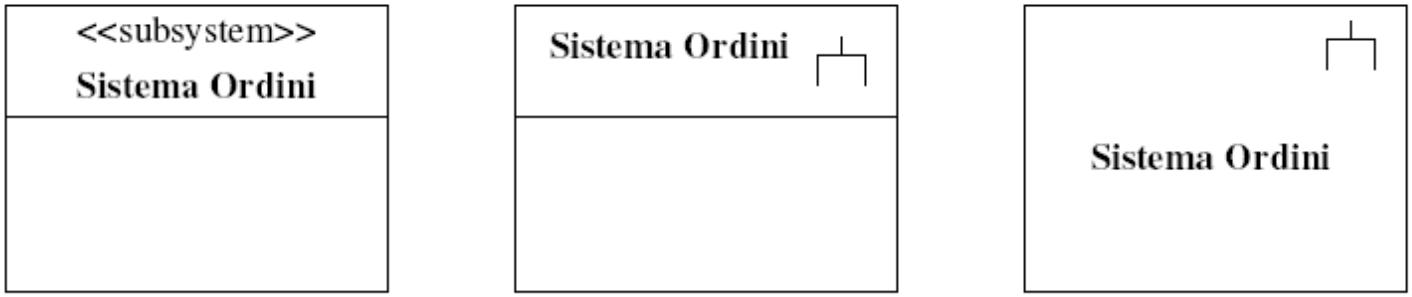
- UML 2.0 permette di connettere alla stessa porta più interfacce



Il componente ha una porta complessa, con interfacce offerte e interfacce richieste

Componenti e sottosistemi

- Mentre in UML 1.x un ***subsystem*** è un tipo di package
- In UML 2 è un tipo di componente
- È quindi possibile specificare per un subsystem
  le interfacce richieste e quelle fornite,
  per evidenziare le relazioni con altri subsystem



Notazioni alternative per il meccanismo di raggruppamento **subsystem**

Diagramma Componenti

- Il diagramma dei componenti deve essere impiegato solamente negli stadi finali della fase di progettazione del sistema
- Tale diagramma rispecchia molto da vicino la struttura che dovrebbe avere il codice e, in un qualche modo, rappresenta l'architettura del sistema
- Potrebbe essere pensato come lo stadio finale dell’evoluzione dell'architettura logica
  che in fase di analisi viene rappresentata
  attraverso il diagramma dei package

Esempio Villaggio Turistico

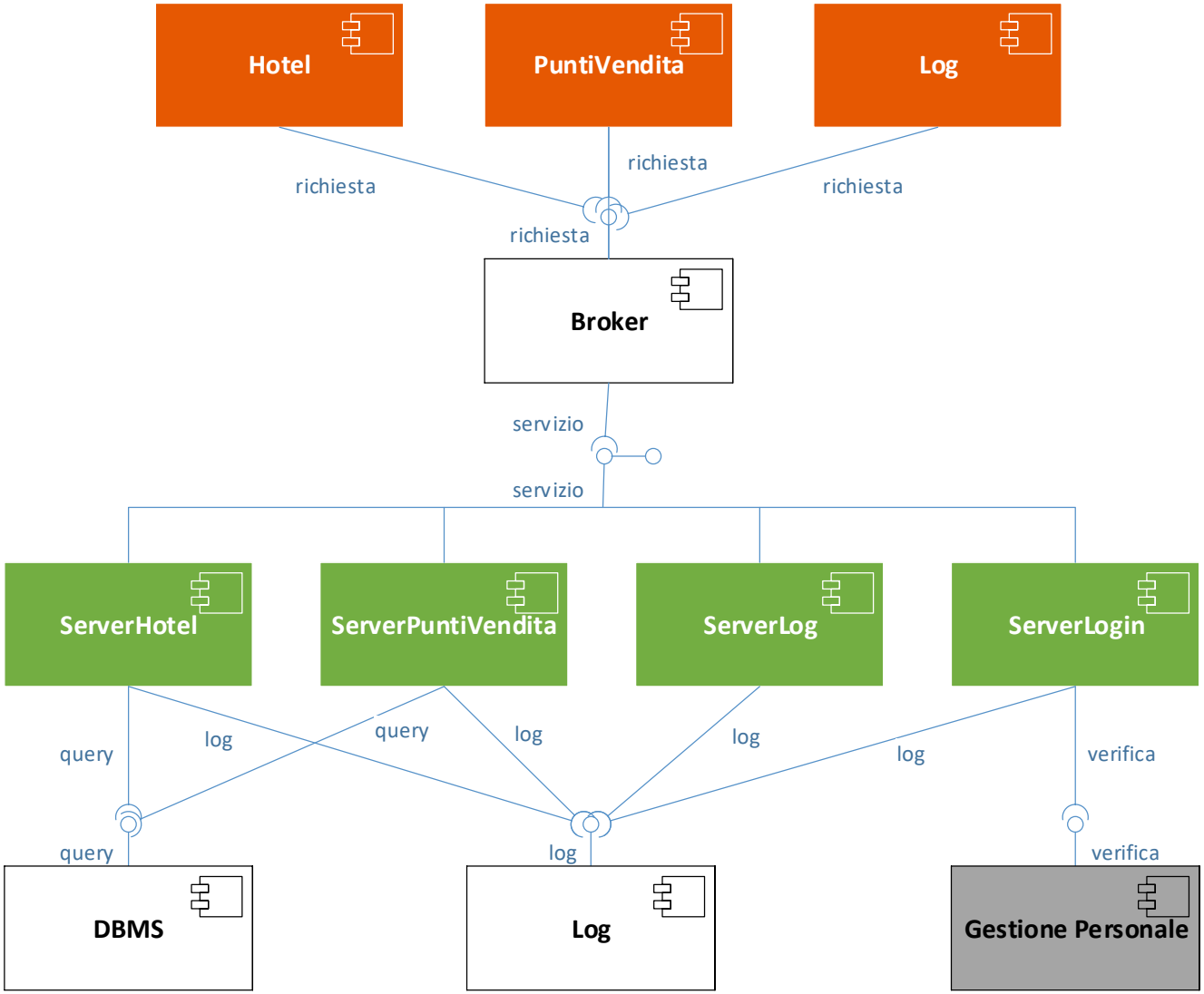


Diagramma di deployment

Diagramma di Deployment

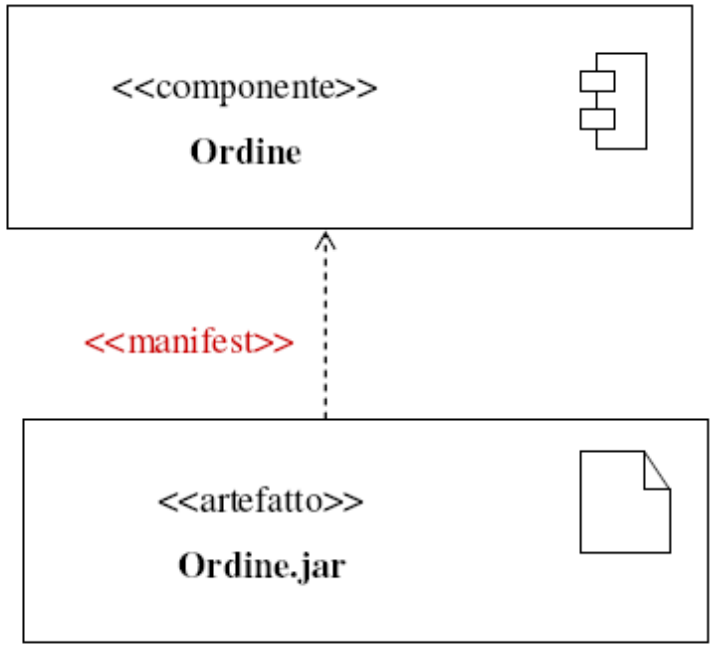
- I diagrammi di deployment ***documentano la distribuzione fisica*** di un sistema, mostrando i vari pezzi di software in esecuzione sulle macchine fisiche
- I diagrammi di deployment quindi mostrano:
  - i collegamenti che permettono la comunicazione fisica tra i pezzi hardware
  - le relazioni tra macchine fisiche e processi software, con l'indicazione dei vari punti in cui viene eseguito il codice

Diagramma di Deployment

- Gli elementi principali del diagramma:
  - Artifact
    - rappresenta una specifica porzione fisica di informazioni utilizzata o prodotta dal processo di sviluppo del software
    - esempi di artifact (manufatti): i modelli (un diagramma dei casi d'uso, un diagramma delle classi, …), file sorgenti, script, file eseguibili, …
    - tipicamente viene utilizzata una relazione di dipendenza <<manifest>> che illustra gli elementi di modellazione (in genere, componenti) utilizzati nella costruzione o generazione di un artefatto
  - Node
    - un'unità sulla quali risiedono e/o sono eseguiti componenti/artefatti
    - i nodi comunicano tra loro tramite CommunicationPath
    - l'allocazione degli artefatti su un nodo viene rappresentata con una relazione di dipendenza <<deploy>> tra il nodo e l'artefatto
  - Device
    - è una risorsa fisica computazionale con capacità elaborative sulla quale possono essere allocati artefatti per l'esecuzione

Diagramma di Deployment

- “Manifest” è la relazione di dipendenza che illustra gli elementi di modellazione utilizzati nella costruzione o generazione di un artefatto



Il componente rappresenta un **tipo di implementazione** fisica, l'artefatto è l'attuale implementazione

In base ai principi di MDA, lo stesso tipo di componente può essere implementato in differenti tecnologie e quindi in differenti artefatti fisici

Esempio Villaggio Turistico

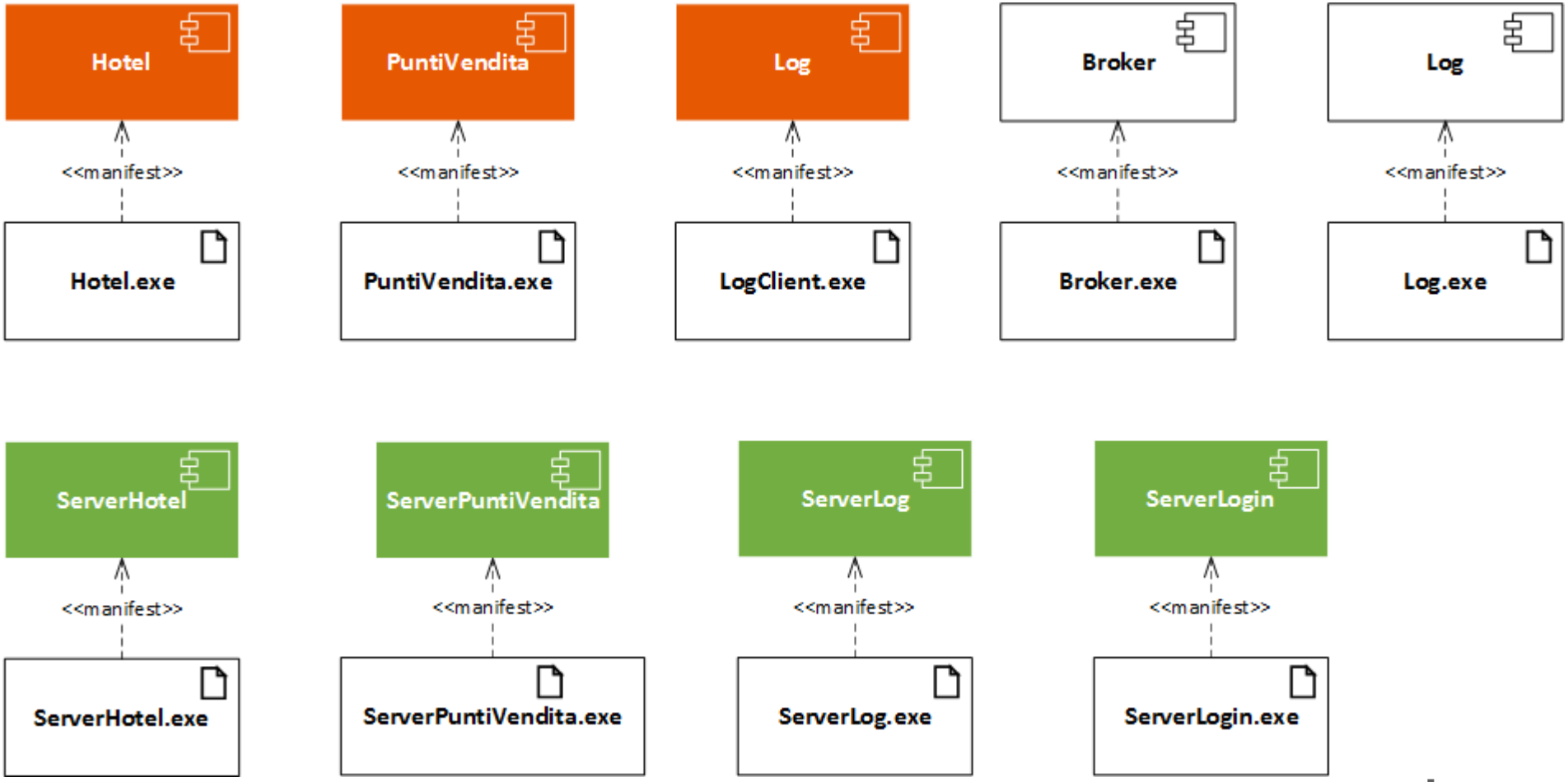
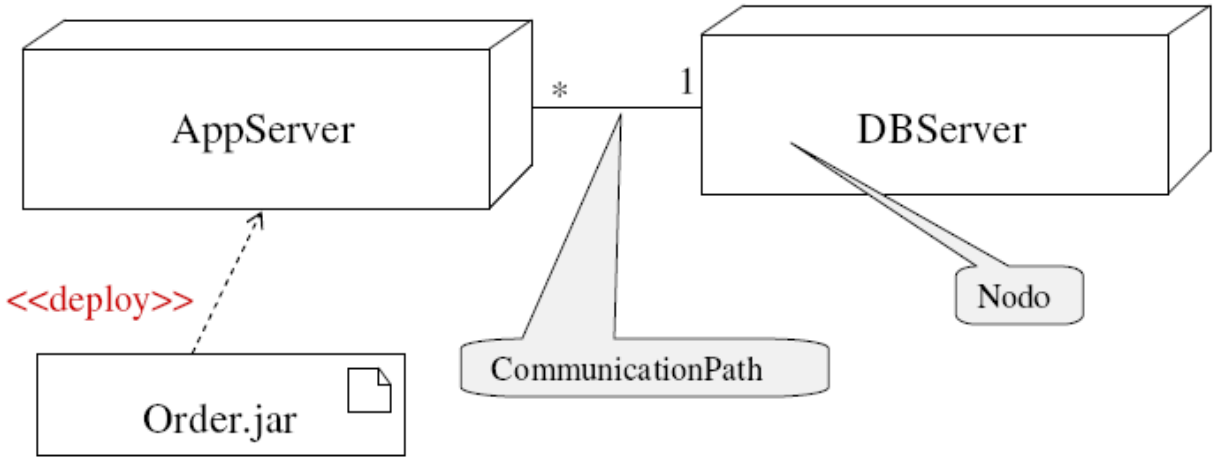


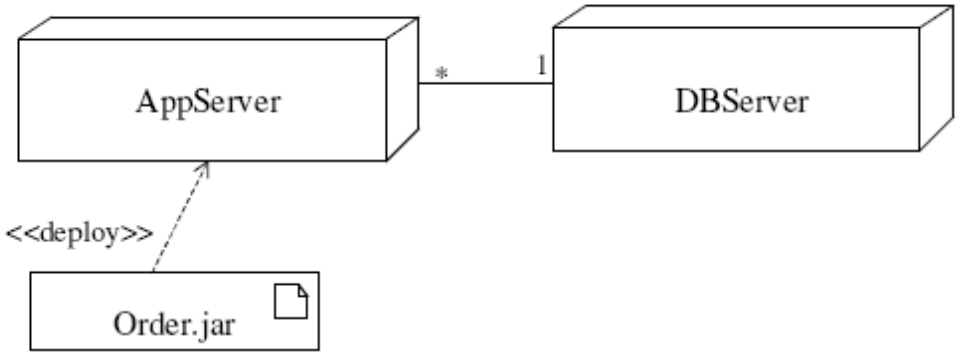
Diagramma di Deployment

- Un **nodo** rappresenta qualsiasi cosa possa eseguire un lavoro: un server, un device o un'unità organizzativa
- È una risorsa su cui gli artefatti possono essere allocati per l'esecuzione, questo fatto viene rappresentato con una dipendenza di tipo <<deploy>> tra il nodo e l'artefatto

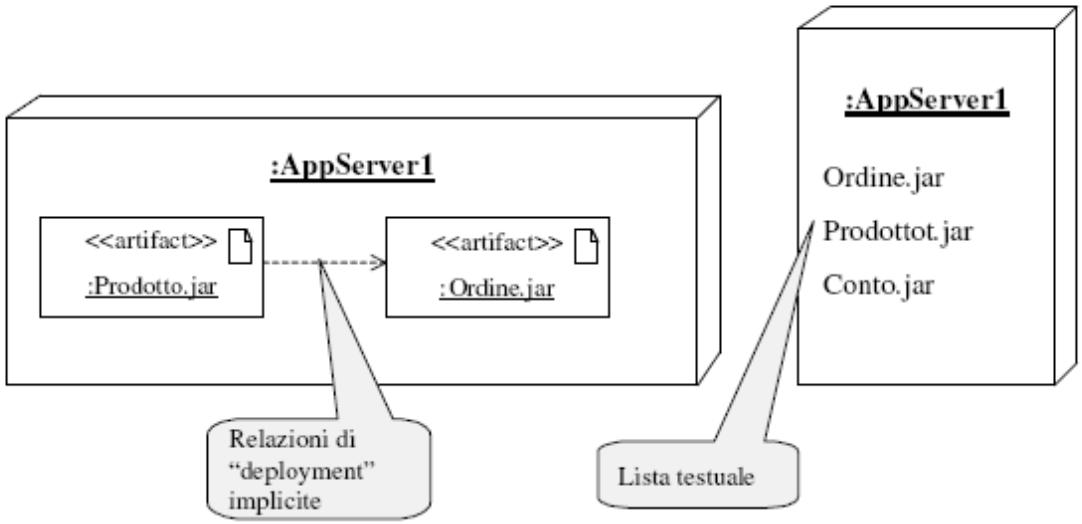


Un communication path è un'associazione tra due nodi tramite la quale i nodi possono scambiarsi segnali e messaggi

Diagramma di Deployment



Livello logico: type-level



Livello fisico: instance-level

Diagramma di Deployment

- L'Execution Environment è un nodo che offre l'ambiente per l'esecuzione di specifici tipi di componenti che sono allocati su di esso
  - <<OS>>
  - <<databasesystem>>
  - <<J2EE container>>
  - …

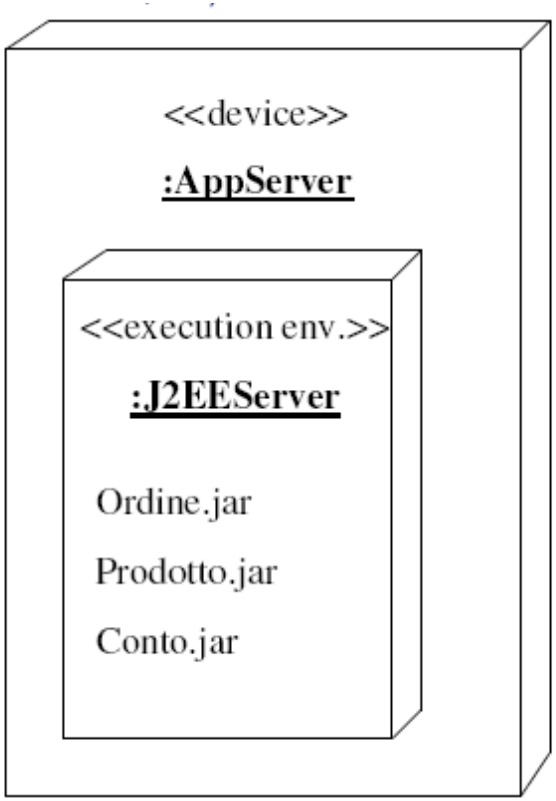
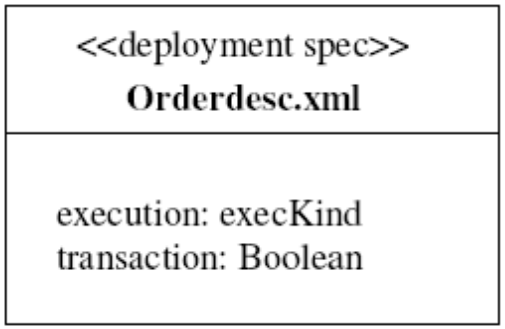
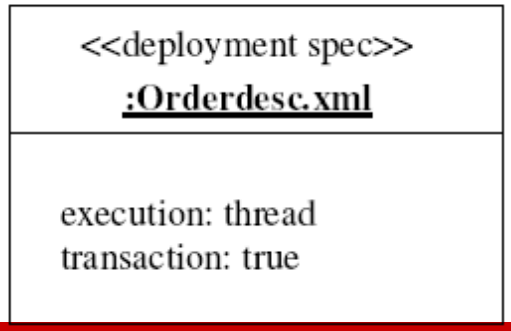


Diagramma di Deployment

- Deployment Specification: è un insieme di proprietà che determinano i parametri di esecuzione di un artefatto allocato su un nodo



Type-level (Specification level)



Instance-level

Esempio Villaggio Turistico